RAG-SR: RETRIEVAL-AUGMENTED GENERATION FOR NEURAL SYMBOLIC REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic regression is a key task in machine learning, aiming to discover mathematical expressions that best describe a dataset. While deep learning has increased interest in using neural networks for symbolic regression, many existing approaches rely on pre-trained models. These models require significant computational resources and struggle with regression tasks involving unseen functions and variables. A pre-training-free paradigm is needed to better integrate with search-based symbolic regression algorithms. To address these limitations, we propose a novel framework for symbolic regression that integrates evolutionary feature construction with a neural network, without the need for pre-training. Our approach adaptively generates symbolic trees that align with the desired semantics in real-time using a language model trained via online supervised learning, providing effective building blocks for feature construction. To mitigate hallucinations from the language model, we design a retrieval-augmented generation mechanism that explicitly leverages searched symbolic expressions. Additionally, we introduce a scale-invariant data augmentation technique that further improves the robustness and generalization of the model. Experimental results demonstrate that our framework achieves state-of-the-art accuracy across 25 regression algorithms and 120 regression tasks [1].

1 INTRODUCTION

Symbolic regression (SR) is a machine learning technique that searches the space of symbolic expressions to identify models that best fit a dataset (Sun et al., 2023; Fong et al., 2023). Unlike traditional regression methods, which assume a fixed model structure, SR automatically determines both the structure and parameters of the model. This flexibility allows SR to achieve both high accuracy and interpretability, making it especially valuable in fields such as physics (Udrescu et al., 2020), biology (Brunton et al., 2016), and finance (Liu & Guo, 2023), where uncovering transparent, understandable models is crucial for scientific discovery and informed decision-making.

In this paper, we focus on an automated feature construction approach to SR (Cava et al., 2019). The key idea is to generate a set of symbolic trees/features, $\Phi = \{\phi_1, \ldots, \phi_m\}$, from a dataset (X, Y) to enhance the performance of an interpretable model $\mathcal{M}$, such as linear regression (Cava et al., 2019; Zhang et al., 2023a). The objective is to minimize the loss function $\mathcal{L}(\Phi; \mathbf{X}, y)$, defined as:

$$\mathcal{L}(\Phi;\mathbf{X}, y) = \frac{1}{N}\sum_{i=1}^{N} \ell\left(\mathcal{M}\left(\phi_1(X_i), \ldots, \phi_m(X_i)\right), y_i\right), \tag{1}$$

where N represents the number of instances, and X_i and y_i represent the features and label for the i-th instance in the training data. By decomposing the SR task into the discovery of feature sets, this approach reduces the complexity of the problem. Even if each feature ϕ is weakly correlated with the target Y, the model can still perform well as long as the features collectively complement each other in predicting the target. This symbolic regression paradigm is particularly effective for complex real-world problems, where the complexity of the underlying system cannot be captured by a simple equation.

---

[1] Source Code: `https://anonymous.4open.science/r/RAG_SR_ICLR_2025/experiment/README-RAG-SR.md`

Traditional SR methods, predominantly based on genetic programming (GP) (Banzhaf et al., 1998), perform gradient-free searches within the symbolic space (Jiang & Xue, 2024). While effective at exploration, these methods often lack search effectiveness due to limited guidance from accumulated knowledge during the evolutionary process. Recent advances in deep learning for SR (Biggio et al., 2021; Kamienny et al., 2022) aim to address these inefficiencies by leveraging knowledge more effectively.

Deep learning-based SR typically follows three primary paradigms: pre-trained language models (Biggio et al., 2021; Kamienny et al., 2022), reinforcement learning (Landajuela et al., 2021), and sparse supervised learning (Sahoo et al., 2018). Sparse supervised learning does not generate symbolic models directly; instead, it relies on heuristic pruning and neural architecture search to sparsify the network so that it can be converted into symbolic expressions (Li et al., 2024). In contrast, pre-trained language models and reinforcement learning can generate symbolic models directly. However, pre-trained language models require prior assumptions about the problem space, limiting their generalizability to novel tasks involving unseen functions and features. Additionally, identifying an optimal set of features for modeling complex real-world systems is time-consuming, making it impractical to generate many pairs of symbolic models and their outputs for pre-training. While reinforcement learning with a language model offers task adaptability (Landajuela et al., 2022), its low sample efficiency remains a significant drawback. Therefore, it is desirable to explore supervised learning methods that do not rely on pretraining for SR to overcome these challenges.

To develop an effective and efficient neural network for SR, we propose a novel neural network-based symbolic regression framework inspired by geometric semantic genetic programming (GSGP) (Moraglio et al., 2012). As illustrated in Figure 1, the core idea is to use a neural network to dynamically predict the best feature ϕ to replace an existing feature in the current set $\Phi = \{\phi_1, \ldots, \phi_m\}$, with the goal of filling the gap in the residual R, referred to as the desired semantics in this paper. Throughout the evolutionary process, the relationship between the semantics/outputs of each symbolic tree $\phi(X)$ and its symbolic representation ϕ is captured and stored in a neural semantic library, which is continuously updated in an online fashion.

One challenge with neural semantic libraries is that language models may generate features ϕ that are grammatically correct but irrelevant to the desired semantics R. In the language model domain, this is known as hallucination (Sun et al., 2024). To mitigate this, we propose a retrieval-augmented generation technique to reduce hallucination and generate symbolic trees that better align with the desired semantics. In summary, the key contributions of this paper are as follows:

- We propose a semantic descent algorithm to optimize symbolic models using a neural network with online supervised learning. The neural network continuously learns to generate symbolic trees that precisely capture the desired semantics, pushing the boundaries of deep symbolic regression to handle complex problems.
- To reduce hallucination in language models, we develop a retrieval-augmented generation mechanism. This technique makes the generated symbolic models are not only grammatically correct but also better aligned with the desired semantics, resulting in more accurate predictions.
- To better capture the relationship between desired semantics and retrieved symbolic expressions, we propose a masked contrastive loss, which more accurately generates symbolic trees by aligning the embeddings of desired semantics with those of the retrieved expressions.
- We propose a data augmentation and double query strategy to fully exploit the scale-invariant properties of feature construction-based symbolic regression, further improving the effectiveness of generated symbolic expressions.

2 RELATED WORK

In the domain of neural symbolic regression, a key advantage of pre-trained models is that, once pre-trained (Biggio et al., 2021; Kamienny et al., 2022), models can be reused for similar tasks without further optimization. These models are designed to solve a distribution of tasks through mechanisms such as invariance encoding (Holt et al., 2023), contrastive learning (Li et al., 2022),

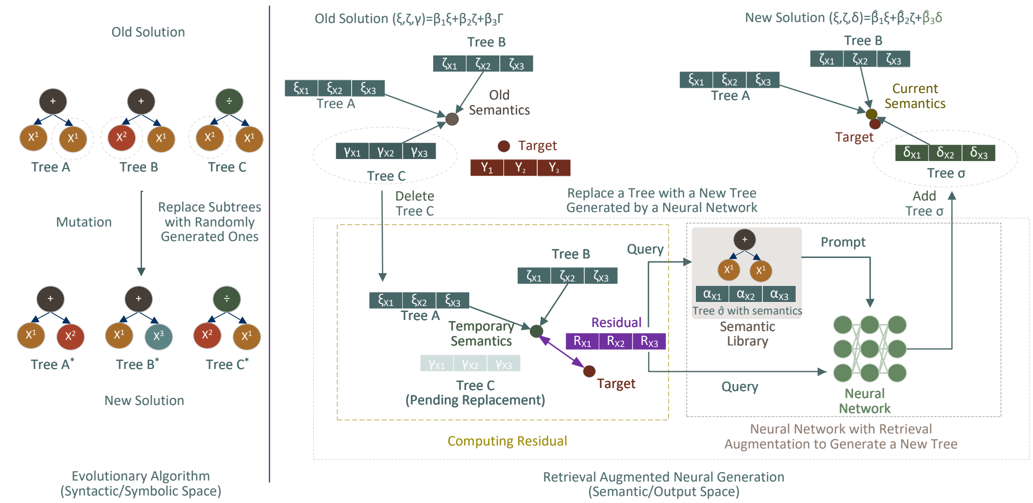

Old Solution
Tree A
Tree B
Tree C
Mutation
Replace Subtrees with Randomly Generated Ones
Tree A*
Tree B*
Tree C*
New Solution
Evolutionary Algorithm (Syntactic/Symbolic Space)
Old Solution (ξ,ζ,γ)=β1ξ+β2ζ+β3Γ
Old Semantics
Target
Delete Tree C
Replace a Tree with a New Tree Generated by a Neural Network
New Solution (ξ,ζ,δ)=β1ξ+β2ζ+β3δ
Current Semantics
Tree σ
Add Tree σ
Query
Prompt
Tree δ with semantics
Semantic Library
Residual
Temporary Semantics
Tree C (Pending Replacement)
Neural Network
Computing Residual
Neural Network with Retrieval Augmentation to Generate a New Tree
Retrieval Augmented Neural Generation (Semantic/Output Space)

Figure 1: Comparison of the evolutionary algorithm and retrieval-augmented neural semantic library for feature construction-based symbolic regression.

or conditional constraints (Bendinelli et al., 2023) to capture relationships among different SR tasks within a problem space. However, these methods may struggle with tasks beyond the scope of the pre-training data, particularly when encountering different function sets or more variables than those seen during training (Shojaee et al., 2024a; Meidani et al., 2024). Fine-tuning could alleviate the misalignment between training and target tasks, through approaches like reinforcement learning (Holt et al., 2023) or using imitation learning to learn successful mutations (Kamienny et al., 2023). However, fine-tuning large pre-trained language models can be challenging. Thus, exploring how online learning techniques can be applied exclusively to enhance SR remains a promising and underexplored direction.

Reinforcement learning (RL), on the other hand, learns the probability distribution of promising symbolic models (Landajuela et al., 2021; Xu et al., 2024) by interacting with the environment, allowing it to adapt to different function sets for various tasks. However, deep symbolic optimization via RL often suffers from low sample efficiency, requiring integration with GP (Mundhenk et al., 2021) or Monte Carlo Tree Search (MCTS) (Xu et al., 2024) techniques to improve performance. Furthermore, RL typically simplifies feedback to a scalar reward, such as mean squared error (Landajuela et al., 2021), which limits the richness of information provided during the search process. A more effective approach would involve using a loss vector rather than a scalar loss to provide richer feedback and enhance overall search effectiveness.

Sparse supervised learning methods, such as deep equation learners (Sahoo et al., 2018) and efficient symbolic policy learning (Guo et al., 2024), aim to derive interpretable symbolic models by regularizing neural networks (Zhang et al., 2023c). However, since the L_0 norm is non-differentiable, these techniques often rely on heuristic pruning approaches to convert neural networks into interpretable expressions. Additionally, they typically require neural architecture search methods to identify suitable architectures before gradient-based training (Li et al., 2024).

Evolutionary symbolic regression is primarily based on the GP framework, which automatically discovers symbolic models without predefined structures to fit the training data (Fong et al., 2023). Recently, semantic GP has gained substantial attention (Moraglio et al., 2012; Zhang et al., 2023b). Unlike traditional GP, which operates in the syntactic/symbolic space, semantic GP works in the semantic/output space. By focusing on semantic space, solution generation operators can ensure that the newly generated solutions have more predictable behavior, such as guaranteed loss reduction—something conventional GP operators often lack. A key challenge in semantic GP is generating GP trees that satisfy the desired semantics (Moraglio et al., 2012). A common strategy is to build a semantic library that stores evaluated GP trees (Pawlak et al., 2014). In semantic mutation, this library is searched for trees that closely match the target semantics, and the best-matching tree is selected. However, this approach relies solely on existing building blocks without leveraging historical knowledge to create new ones. To address this issue, it is crucial to incorporate deep learning

Algorithm 1 Semantic Descent

```
1: Input: Features $\Phi = \{\phi_1, \dots, \phi_m\}$, semantics library $\mathcal{L}$, neural network model $\mathcal{N}$, current semantics
   $\Phi(X)$, target $Y$, neural generation probability $P_{\text{neural}}$
2: Output: Updated features $\Phi$
3: $\mathcal{O} \leftarrow$ Random permutation of $\{1, 2, \dots, m\}$                  ▷ Shuffle tree indices
4: for each $i \in \mathcal{O}$ do
5:     $\tilde{\phi}_i(X) \leftarrow \frac{\phi_i(X) - \mu_i}{\sigma_i}$                  ▷ Normalized feature
6:     $\Phi(X)^{\text{temp}} \leftarrow \Phi(X) - \beta_i \tilde{\phi}_i(X)$
7:     $\mathbf{R} \leftarrow Y - \Phi(X)^{\text{temp}}$                  ▷ Compute residual $R$
8:     if rand() $< P_{\text{neural}}$ then
9:         $\phi_i \leftarrow \mathcal{N}(\mathbf{R}, \mathcal{L})$                  ▷ Generate new tree using neural model
10:        $\Phi(X) \leftarrow \Phi(X)^{\text{temp}}$
11:        continue                  ▷ Proceed to next tree
12:    end if
13:    $\phi_{\text{new}} \leftarrow \texttt{ExactRetrieval}(\mathbf{R}, \phi_i, \mathcal{L})$
14:    $\phi_i, \Phi(X) \leftarrow \texttt{ExactReplacement}(\phi_{\text{new}}, \phi_{\text{new}}(X), \Phi(X), \Phi(X)^{\text{temp}}, \mathbf{R}, Y)$
15: end for
```

techniques to learn from the evolutionary learning process and generate better symbolic models that align with the desired semantics.

3 ALGORITHM

The proposed method is based on an evolutionary algorithm framework, encompassing solution initialization, generation, evaluation, selection, and archive maintenance. This work primarily focuses on the solution generation phase, introducing a neural semantic library for solution generation, designed to explicitly retain and apply knowledge throughout the evolutionary process. Solution generation consists of two primary components: semantic descent and retrieval-augmented generation.

3.1 SEMANTIC DESCENT

In this work, we propose Semantic Descent (SD), an iterative optimization procedure designed to improve model performance by selectively replacing suboptimal features. Unlike methods such as geometric semantic GP (Moraglio et al., 2012) or gradient boosting (Feng et al., 2018), which incrementally add new features to minimize error, SD focuses on replacing existing trees in the model with more informative ones. This approach helps maintain a compact model structure while continuously improving accuracy.

At each iteration, a tree ϕ_i is randomly selected from the set of trees $\{\phi_1, \dots, \phi_m\}$ that define the semantics/outputs of the model $\Phi(X) = \beta_1\phi_1(X) + \cdots + \beta_m\phi_m(X) + \alpha$, where β represents the coefficients and α is the intercept. The contribution of ϕ_i is temporarily removed, resulting in temporary semantics $\Phi^{\text{temp}}(X) = \Phi(X) - \beta_i\phi_i(X)$. The residual $R = Y - \Phi^{\text{temp}}(X)$ of the model is then computed, where Y is the target output. The residual R represents the difference between the prediction and the target.

As shown in Algorithm 1, the core idea of SD is to fill the gap in the residual R by replacing the current tree ϕ_i with a better alternative, either generated by a neural model $\mathcal{N}$ (line 9) or retrieved from a semantic library $\mathcal{L}$ (line 13). The semantic library $\mathcal{L}$ stores all previously evaluated symbolic trees and subtrees ψ along with their semantics/outputs $\psi(X)$. The neural model $\mathcal{N}$ learns the mapping between the semantics $\psi(X)$ and the corresponding symbolic tree ψ. This enables the neural network $\mathcal{N}$ to construct a new feature ϕ_{new} using R as input, thereby generating a new feature to reduce the model's error.

The probability of generating new trees using the neural network is P_{neural}, detailed in Section 3.2. The probability of retrieving a tree from the semantic library is $1 - P_{\text{neural}}$. The key idea of exact retrieval is to search the library for the tree that most closely matches the desired semantics, i.e., the residual R, as detailed in Appendix C. Since the linear regression model automatically adjusts feature magnitudes and intercepts, the residual R is normalized using the L_2 norm before being used as input for retrieval or neural generation, i.e., $R \leftarrow \frac{R - \bar{R}}{||R||_2}$. The replacement process is repeated iteratively until all trees ϕ within the solution Φ have been traversed. By focusing on feature replacement instead of addition, SD enables efficient model refinement while maintaining a fixed feature set, allowing for both interpretability and performance improvements.

3.2 RETRIEVAL-AUGMENTED GENERATION

To learn the mapping between symbolic trees, ϕ, and their corresponding semantics, $\phi(X)$, the process involves three steps: First, the trees and semantics are collected from the evolutionary process (Section 3.2.2). Next, they are converted into training data using specially designed encoding rules (Section 3.2.1). Finally, a neural network is trained on the collected data (Section 3.2.3), using cross-entropy loss and masked contrastive loss (Section 3.2.4).

3.2.1 DATA COLLECTION AND NETWORK TRAINING

The semantic library $\mathcal{L}$ is dynamically constructed during the evolutionary process. During solution evaluation, each subtree ψ and its corresponding semantics $\psi(X)$ are stored in a first-in-first-out queue Q with an upper limit of 10,000 entries for training the neural network and future retrieval. To facilitate efficient retrieval, a k-dimensional tree (k-d tree) is constructed using the semantics stored in Q at the end of each generation in the evolutionary process, reducing query complexity to $O(\log(N))$, where N is the number of stored trees. The neural network is also trained at the end of each generation. To prevent unnecessary training, an internal validation set monitors performance degradation. If the validation loss does not increase, network training is skipped for that generation to save computational resources. Nevertheless, the retrieval library is updated even when network training is bypassed, ensuring that knowledge base is continuously updated throughout the evolutionary process.

3.2.2 ENCODING AND DECODING RULES FOR SYMBOLIC TREES

To ensure that the generated symbolic expression is always valid and to eliminate the need for an end token in the language model, we designed a specialized encoding and decoding scheme. Symbolic trees are encoded using a level-order traversal method, specifically through breadth-first search, to convert the tree into a linear sequence. To maintain interpretability, the number of functions in the symbolic tree is capped at n_F. Given this limit and the maximum number of children any function can have, known as maximum arity, α_{max}, the number of terminals n_T required to fill the symbolic tree in the worst case is:

$$n_T = 1 + (n_F \times (\alpha_{\text{max}} - 1)). \tag{2}$$

Given n_F and n_T, the output of the neural network is structured as a fixed-length sequence consisting of n_F functions or terminals, followed by n_T terminals. This structure transforms the symbolic tree generation task into a multi-class classification problem. The first n_F elements of the sequence are decoded into either functions or terminals, while the subsequent n_T elements are restricted to terminals by setting the probability of selecting a function to zero. Detailed pseudocode for the encoding and decoding processes is provided in Appendix E.

3.2.3 OVERALL ARCHITECTURE FOR RETRIEVAL-AUGMENTED GENERATION

As shown in Figure 2, the neural architecture consists of two main components: a Multilayer Perceptron (MLP) and a Transformer model. Their relationship is defined as:

$$\mathbf{O} \in \mathbb{R}^{B \times L \times S} = \text{Transformer Decoder}\Big(\text{Transformer Encoder}(\hat{\phi}) \oplus \text{MLP}(R)\Big) \cdot \mathbf{W}^T. \tag{3}$$

Each of the two components plays a distinct and complementary role in generating a symbolic tree ϕ based on the desired semantics R. The MLP transforms raw semantics into a meaningful feature representation that can guide the generation of the symbolic tree. On the other hand, the Transformer encoder processes the nearest symbolic tree $\hat{\phi}$, retrieved from the semantics library $\mathcal{L}$, which serves as a prompt to reduce hallucination. The outputs of the MLP and Transformer are concatenated and then passed through a Transformer decoder to generate a sequence of L tokens. $\mathbf{W} \in \mathbb{R}^{S \times D}$ is a linear layer that projects the output of the Transformer decoder $\mathbf{H}_{\text{Decoder}} \in \mathbb{R}^{B \times L \times D}$ into the symbol space $\mathbf{O} \in \mathbb{R}^{B \times L \times S}$, where S is the number of unique symbols. These tokens are subsequently decoded into a valid symbolic tree.

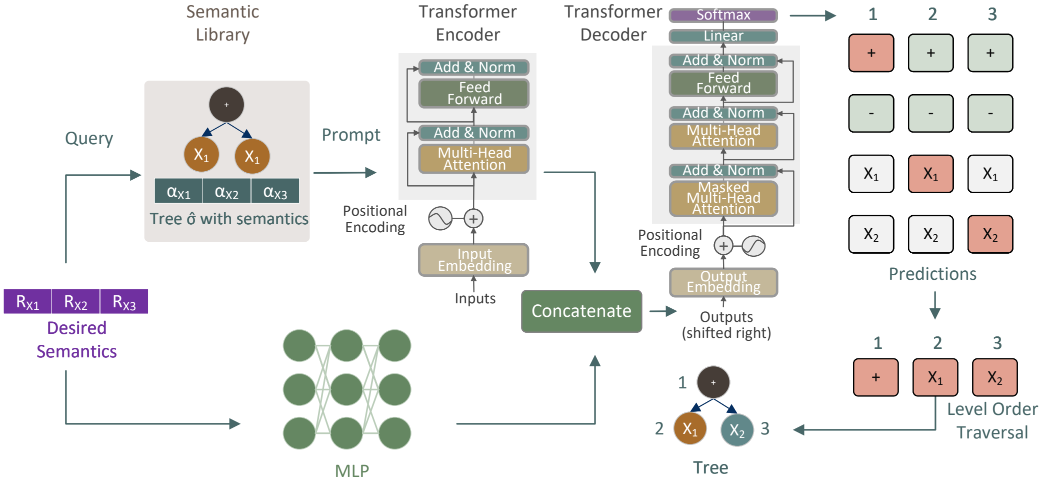


Figure 2: Neural network architecture for symbolic tree generation.

Intention Encoding: The desired semantics $R \in \mathbb{R}^{B \times N}$ is processed through an MLP to produce a feature matrix $\mathbf{F}_{\text{MLP}} \in \mathbb{R}^{B \times K}$, where K is the dimensionality of the hidden layer. The MLP consists of N_L layers, and at each layer i, the transformation is defined as:

$$\mathbf{x}_{i+1} = \text{Dropout}_i\left(\text{SiLU}_i\left(\text{BN}_i\left(\mathbf{W}_i \cdot \mathbf{x}_i + \mathbf{b}_i\right)\right)\right) + \mathbf{x}_i, \tag{4}$$

where $\mathbf{x}_i \in \mathbb{R}^{B \times K}$ is the input to the i-th layer, $\mathbf{W}_i \in \mathbb{R}^{K \times K}$ is the weight matrix, $\mathbf{b}_i \in \mathbb{R}^{K}$ is the bias vector, BN_i denotes the batch normalization layer, SiLU_i is the Sigmoid Linear Unit activation function (Elfwing et al., 2018), and Dropout_i is the dropout layer with a specified dropout rate. This MLP layer results in a feature matrix $\mathbf{F}_{\text{MLP}} \in \mathbb{R}^{B \times K}$, which is then passed through a linear layer to match the dimensionality from K to D, yielding $\mathbf{F}_{\text{MLP}}^{\text{mapped}} \in \mathbb{R}^{B \times D}$, where D is the dimensionality of the Transformer-encoded representation.

Retrieval-Augmented Encoding: For the desired semantics R, a KD-Tree is used to retrieve the nearest symbolic tree $\hat{\phi}$ from the semantic library $\mathcal{L}$, based on Euclidean distance and subject to the constraint that the tree contains no more than n_F nodes. The retrieved tree $\hat{\phi}$ is then processed through an embedding layer to generate $\mathbf{V}_{\hat{\phi}} \in \mathbb{R}^{B \times L \times E}$, where L is the sequence length of the tree encoding and E is the dimensionality of the embedding space. The embedding layer consists of an embedding matrix $\mathbf{E} \in \mathbb{R}^{S \times E}$. The embedded representation $\mathbf{V}_{\hat{\phi}}$ is then encoded using the Transformer model to produce a symbolic model embedding $\mathbf{H}_{\text{Transformer}} \in \mathbb{R}^{B \times L \times D}$. The Transformer encoder applies self-attention and feedforward layers with residual connections as follows:

$$\begin{aligned}
\mathbf{H}_{\text{Self-Attn}} &= \text{LayerNorm}\left(\mathbf{V}_{\hat{\phi}} + \text{SelfAttention}\left(\mathbf{V}_{\hat{\phi}}\right)\right) \quad \in \mathbb{R}^{B \times L \times K}, \\
\mathbf{H}_{\text{Transformer}} &= \text{LayerNorm}\left(\mathbf{H}_{\text{Self-Attn}} + \text{FeedForward}\left(\mathbf{H}_{\text{Self-Attn}}\right)\right) \quad \in \mathbb{R}^{B \times L \times D}.
\end{aligned} \tag{5}$$

Decoding: The combined feature representation $\mathbf{H}_{\text{Combined}} = \mathbf{F}_{\text{MLP}}^{\text{mapped}} \oplus \mathbf{H}_{\text{Transformer}} \in \mathbb{R}^{B \times (L+1) \times D}$ is fed into a Transformer decoder to generate the contextual embeddings $\mathbf{H}_{\text{Decoder}} \in \mathbb{R}^{B \times L \times D}$. The decoding process is performed auto-regressively, utilizing a greedy decoding strategy.

3.2.4 LOSS FUNCTION

Masked Contrastive Loss: The intention encoding should ideally learn useful knowledge not only from target expressions ϕ but also from the retrieved symbolic expressions $\hat{\phi}$. In parallel, the retrieval-augmented encoding should be aware of the semantics of the nearest symbolic expressions. To fulfill these objectives, we propose a contrastive loss that aligns the embeddings from both the intention encoding and the retrieval-augmented encoding components.

Given the nearest semantics $\hat{\phi}(X) \in \mathbb{R}^{B \times N}$, it is processed through a MLP to generate a feature matrix of nearest semantics $\mathbf{F}_{\text{nearest}} \in \mathbb{R}^{B \times K}$. Simultaneously, the embedding of the symbolic model

$\mathbf{H}_{\text{Transformer}} \in \mathbb{R}^{B\times L\times D}$ is averaged along the sequence length dimension to produce the averaged embedding $\mathbf{H}_{\text{avg}} \in \mathbb{R}^{B\times D}$. Then, the InfoNCE loss (Oord et al., 2018), a popular objective in contrastive learning, is employed to maximize the similarity between the nearest semantics feature matrix $\mathbf{F}_{\text{nearest}}$ and the averaged symbolic embeddings $\mathbf{H}_{\text{avg}}$, while minimizing similarity with negative samples from the same batch. To alleviate false negatives, i.e., when two samples in a batch are semantically similar, the InfoNCE loss is masked by a mask matrix $mask$. The mask matrix is designed such that non-diagonal elements with an absolute cosine similarity greater than 0.99 are marked as false (indicating false negatives), while all other entries are marked as true. The masked InfoNCE loss is formally defined as:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp\left(\text{sim}(\mathbf{F}_{\text{nearest}}[i], \mathbf{H}_{\text{avg}}[i])/\tau\right)}{\sum_{j=1}^{B}\exp\left(\text{sim}(\mathbf{F}_{\text{nearest}}[i], \mathbf{H}_{\text{avg}}[j])\cdot\text{mask}/\tau\right)}, \tag{6}$$

where $\text{sim}(\cdot,\cdot)$ denotes cosine similarity, and τ is a temperature parameter controlling the sharpness of the softmax function. This contrastive loss ensures that the nearest semantics are closely aligned with their corresponding symbolic representations in the embedding space, while differentiating them from unrelated samples.

Cross-Entropy Loss: The model is also trained using cross-entropy loss over the sequence of L symbols. Let $\mathbf{o}^i_{\text{true}} \in \mathbb{R}^S$ denote the one-hot encoded ground truth for the i-th position, and $\mathbf{o}^i_{\text{pred}} \in \mathbb{R}^S$ denote the predicted probability distribution at that position. Formally, the cross-entropy loss for each sequence is defined as $\mathcal{L}_{\text{cross-entropy}} = -\sum_{i=1}^{L}\mathbf{o}^i_{\text{true}}\cdot\log(\mathbf{o}^i_{\text{pred}})$. The final loss $\mathcal{L}$ is a weighted sum of the cross-entropy loss and the contrastive loss:

$$\mathcal{L} = \mathcal{L}_{\text{cross-entropy}} + \lambda\cdot\mathcal{L}_{\text{InfoNCE}}, \tag{7}$$

where λ is a hyperparameter that balances the contributions of the two losses.

3.3 DATA AUGMENTATION AND DOUBLE QUERY

In linear regression, the sign of coefficients is automatically adjusted, so the sign of the semantics is not crucial. However, the training data may only include one side of a training pair $(\psi, \psi(X))$, without considering its opposite, $(\psi, -\psi(X))$. Consequently, when the desired semantics is $-\psi(X)$, the model may fail to generate the correct symbolic tree ψ. To address this issue, we augment the training data by including both $(\psi, \psi(X))$ and $(\psi, -\psi(X))$ pairs:

$$\mathcal{T} \leftarrow \mathcal{T} \cup \{(\psi, -\psi(X)) \mid (\psi, \psi(X)) \in \mathcal{T}\}. \tag{8}$$

During decoding, both R and $-R$ are used to query the neural network, generating candidate trees ϕ and ϕ'. The tree with the highest probability is selected as the final symbolic model. This technique, referred to as double query (DQ), allows the model to generate symbolic trees with sign-insensitive semantics, thereby improving the effectiveness of neural generation.

4 EXPERIMENTS

This section is divided into two parts. The first part evaluates the effectiveness of the proposed components in improving the prediction accuracy of the neural semantic library. The second part investigates the performance of integrating the SR method with the retrieval-augmented neural semantic library. It compares this integrated approach to state-of-the-art SR methods.

4.1 EXPERIMENTAL RESULTS OF NEURAL SEMANTIC LIBRARY

Experimental Settings: To evaluate the effectiveness of the proposed techniques in enhancing the learning capabilities of the neural semantic library, we conduct the first experiment on synthetic data. The objective is to evaluate how various components contribute to the learning effectiveness of the neural semantic library. In this experiment, 10 variables and 50 training instances are randomly drawn from a Gaussian distribution $\mathcal{N}(0, 100)$. Then, a total of 10000 symbolic expressions with random heights $h \in [0, 5]$ are generated using the grow method (Banzhaf et al., 1998) from GP and evaluated on the randomly generated data. The maximum number of functions n_F is set to 5, and expressions exceeding this limit are filtered out. To avoid redundancy, only one semantically equivalent GP tree is retained, ensuring no symbolic expressions overlap between training and test sets. This setup ensures that the final metric reflects the ability of the neural network to learn patterns and generalize to unseen data, rather than simply fitting to previously seen examples. A total of 80% of the symbolic models are used for training, while the remaining 20% are reserved for testing. The evaluation metric is the edit distance (Matsubara et al., 2022; Bertschinger et al., 2023) between the generated symbolic tree and the ground truth, where a smaller distance indicates that the neural semantic library generates more effective building blocks, significantly aiding the evolutionary algorithm in finding optimal solutions. Each experiment is run 5 times to ensure stable and reliable results.

Parameter Settings: For the neural network, the dropout rate is set to 0.1. The MLP consists of 3 layers, while both the encoder and decoder Transformers have 1 layer each. The hidden layer size is set to 64 neurons. A learning rate of 0.01 and a batch size of 64 are used. Early stopping with a patience of 5 epochs is employed to prevent overfitting. The weight of contrastive loss λ is set to 0.05.

Experimental Results (Edit Distance): The experimental results for edit distance on the test set are presented in Figure 3. First, comparing neural generation with simple retrieval from the library (W/O NN), neural generation performs better by a large margin, indicating the effectiveness of using a neural network for symbolic tree generation. As for the ablation results of components, the results show that including all components achieves the lowest median edit distance, indicating that the combination of all proposed techniques provides the best overall performance. Among the components, the RAG technique has the most significant impact, highlighting that external knowledge from the semantic library significantly improves the neural network's ability to generate relevant symbolic trees. Data augmentation (DA) also plays a crucial role, ranking as the second most important component. Without DA, the model struggles to handle the scale-invariant nature of feature construction, leading to worse performance. The compact boxplots reflect the consistency and reliability of these components. Dropout has a moderate positive effect, indicating that overfitting control techniques are helpful for training the neural semantic library. Similarly, contrastive learning (CL) shows a moderate impact, confirming the effectiveness of using contrastive loss to align the intention encoding with retrieval augmentation encoding components. Finally, DQ also improves effectiveness, showing that even simply generating multiple solutions during inference can lead to better solutions, which aligns with findings from large language models (Wang et al., 2023). The impact of DQ becomes more pronounced in the absence of DA, suggesting that DA partially compensates for the lack of DQ.

Experimental Results (Running Time): The running time comparisons in Figure 4 demonstrate that RAG moderately increases the overall running time. However, one advantage of incorporating RAG into the component is that new trees can be seamlessly added to the retrieval library to improve accuracy without requiring model fine-tuning, making the algorithm efficient for application in an online learning setting. For DA and DQ, removing these components reduces the running time from 44 seconds to 35 and 29 seconds, respectively, indicating that they do introduce some computational overhead. However, given the accuracy improvements they provide, the increase in computational time is acceptable. Although removing both DA and DQ significantly reduces computational cost, the substantial loss of edit distance from 3.82 to 4.35 outweighs the benefit of faster execution.

Examples of Generated Trees: Table 1 provides examples of symbolic trees generated by the neural network with and without retrieval augmentation, along with the retrieved trees. The results demonstrate that the retrieved trees share certain similarities with the ground truth, such as variable usage. These results validate that providing the retrieval tree as a prompt helps the neural network generate more relevant trees, reducing hallucination compared to relying solely on the desired semantics.

4.2 EXPERIMENTS OF RAG-SR

Datasets: In this study, we primarily focus on 120 black-box datasets from the PMLB benchmark (Olson et al., 2017), which are particularly challenging for pre-training methods (Kamienny et al., 2022) due to the potential absence of simple symbolic expressions to model these datasets. The results on the 119 Feynman and 14 Strogatz datasets are presented in Appendix L.2.

Table 1: Examples of symbolic trees generated by the retrieval-augmented neural network, simple neural network, retrieval library, and ground truth.

RAG-NN Generated Tree (Distance)	Simple NN Generated Tree (Distance)
sin(sin(ARG3)) (0)	cos(cos(cos(ARG9))) (4)
aq(ARG7, ARG8) (0)	abs(maximum(ARG7, ARG7)) (3)
maximum(ARG1, ARG8) (0)	subtract(ARG1, ARG1) (2)
sqrt(sqrt(ARG2)) (0)	abs(abs(ARG2)) (2)
subtract(ARG6, ARG7) (0)	maximum(ARG7, ARG7) (2)

Retrieval Tree (Distance)	Ground Truth Tree
sin(ARG3) (1)	sin(sin(ARG3))
abs(negative(maximum(aq(ARG8, ARG0), aq(ARG7, ARG8)))) (6)	aq(ARG7, ARG8)
maximum(add(absolute(sin(cos(ARG6))), ARG8), ARG1) (6)	maximum(ARG1, ARG8)
square(abs(ARG2)) (2)	sqrt(sqrt(ARG2))
subtract(ARG7, ARG6) (2)	subtract(ARG6, ARG7)

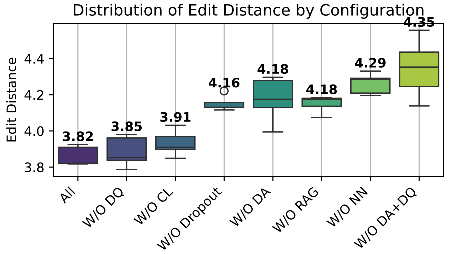


Figure 3: Ablation study of components based on edit distance on the test set.

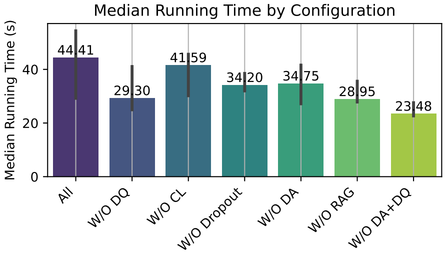


Figure 4: Ablation study of components with respect to running time (training and inference).

Evaluation Protocol: The evaluation follows the established procedures of state-of-the-art symbolic regression benchmarks (La Cava et al., 2021). Specifically, each dataset is split into training and testing sets with a 75:25 ratio, and experiments are repeated 10 times for robustness. The R^2 score on the test set is used as the evaluation metric. To better handle categorical variables, we use a target encoder (Micci-Barreca, 2001). Furthermore, to prevent any single feature from disproportionately influencing the semantics, all input features are normalized using min-max scaling (Raymond et al., 2020).

Parameter Settings: For GP, we follow conventional parameter settings: a population size of 200 and a maximum of 100 generations. Each solution consists of 10 trees, representing 10 features. The probability of using neural generation, P_{neural}, is set to 0.1.

Experimental Results (Accuracy): The experimental results on SRBench are presented in Figure 5. The proposed method, RAG-SR, outperforms all state-of-the-art symbolic regression and machine learning techniques in terms of R^2 scores. Notably, it surpasses the TPSR method (Shojaee et al., 2024a), which combines MCTS with a pre-trained end-to-end Transformer (Kamienny et al., 2022). The improvement is statistically significant, as confirmed by the Wilcoxon signed-rank test with Benjamini-Hochberg correction, shown in Figure 6. This indicates the effectiveness of using a purely online training language model for learning symbolic expressions. Compared to SBP-GP (Pawlak et al., 2014), which is a purely retrieval-based geometric semantic GP that does not use a neural network, the significant advantage of RAG-SR demonstrates the effectiveness of using a neural network to dynamically generate symbolic models.

Experimental Results (Complexity): The model complexity of RAG-SR follows the definition of SRBench, where the final model is converted into a SymPy-compatible expression, and the number of nodes in the symbolic tree is counted as a measure of complexity. As shown in Figure 5, RAG-SR produces models that are an order of magnitude smaller in size compared to PS-Tree (Zhang et al., 2022), which is a piecewise SR method that ranks second in R^2 scores in Figure 5. The Pareto front of test R^2 scores and model size rank is shown in Figure 7, where RAG-SR appears on the first Pareto front, indicating that RAG-SR achieves a good balance between accuracy and model complexity.

Experimental Results (Training Time): The training time of RAG-SR is comparable to that of FEAT, a standard feature-construction-based SR method (Cava et al., 2019), suggesting that the

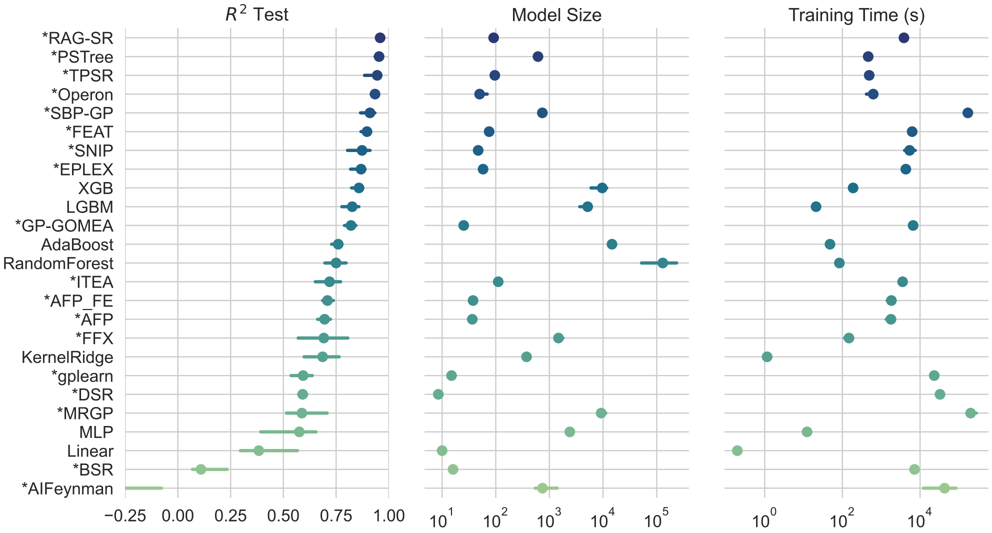
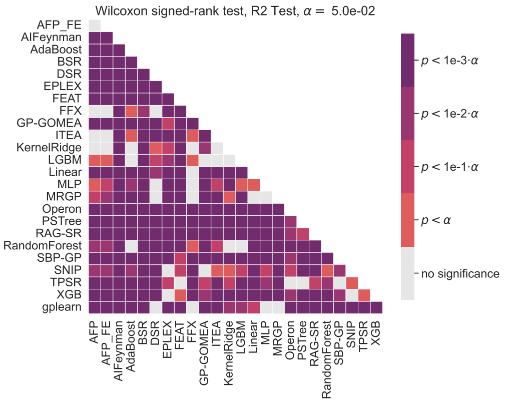


Figure 5: R^2 scores, model sizes, and training time of 25 algorithms on 120 regression problems.



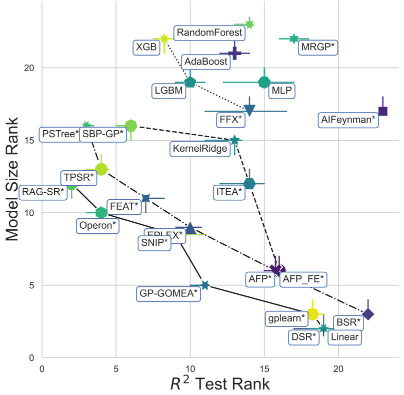


Figure 6: Pairwise statistical comparisons of test R^2 scores on regression problems.

Figure 7: Pareto front of the rank of test R^2 scores and model size for different algorithms.

computational cost of learning a neural semantic library is within an acceptable range. However, compared to TPSR, which directly leverages a pre-trained model to guide SR without requiring fine-tuning, RAG-SR is an order of magnitude slower. This discrepancy is partly due to the fact that, in the current implementation, all neural networks in RAG-SR are trained on a CPU due to limited computational resources. Training the neural networks on a GPU could potentially reduce the computational time of RAG-SR.

5 CONCLUSIONS

In this paper, we propose a novel feature construction-based SR method with a retrieval-augmented neural semantic library. Ablation studies confirm that the retrieval augmentation mechanism effectively mitigates the issue of hallucination, enabling the generation of more accurate symbolic trees that align with the desired symbolic trees. Furthermore, data augmentation and double query techniques effectively improve the neural network's ability to generate symbolic trees that account for the scale-invariant characteristics of feature construction-based SR. Experimental results on large-scale symbolic regression benchmarks demonstrate that RAG-SR significantly outperforms state-of-the-art SR techniques, including those guided by pre-trained language models. For future directions, introducing constraints on model complexity may help reduce the risk of overfitting, particularly with datasets that contain noise or limited samples, presenting a promising direction for future research.

REFERENCES

Ignacio Arnaldo, Krzysztof Krawiec, and Una-May O'Reilly. Multiple regression genetic programming. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 879–886, 2014.

Wolfgang Banzhaf, Peter Nordin, Robert E Keller, and Frank D Francone. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., 1998.

Tommaso Bendinelli, Luca Biggio, and Pierre-Alexandre Kamienny. Controllable neural symbolic regression. In *International Conference on Machine Learning*, pp. 2063–2077. PMLR, 2023.

Amanda Bertschinger, Q Tyrell Davis, James Bagrow, and Joshua Bongard. The metric is the message: Benchmarking challenges for neural symbolic regression. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 161–177. Springer, 2023.

Manish Bhattarai, Javier E Santos, Shawn Jones, Ayan Biswas, Boian Alexandrov, and Daniel O'Malley. Enhancing code translation in language models with few-shot learning via retrieval-augmented generation. *arXiv preprint arXiv:2407.19619*, 2024.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pp. 936–945. Pmlr, 2021.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

William La Cava, Tilak Raj Singh, James Taggart, Srinivas Suri, and Jason Moore. Learning concise representations for regression by evolving networks of trees. In *International Conference on Learning Representations*, 2019.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Ji Feng, Yang Yu, and Zhi-Hua Zhou. Multi-layered gradient boosting decision trees. *Advances in Neural Information Processing Systems*, 31, 2018.

Kei Sen Fong, Shelvia Wongso, and Mehul Motani. Rethinking symbolic regression: Morphology and adaptability in the context of evolutionary algorithms. In *The Eleventh International Conference on Learning Representations*, 2023.

Arya Grayeli, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer, and Swarat Chaudhuri. Symbolic regression with a learned concept library. *arXiv preprint arXiv:2409.09359*, 2024.

Jiaming Guo, Rui Zhang, Shaohui Peng, Qi Yi, Xing Hu, Ruizhi Chen, Zidong Du, Ling Li, Qi Guo, Yunji Chen, et al. Efficient symbolic policy learning with differentiable symbolic expression. *Advances in Neural Information Processing Systems*, 36, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

Thomas Helmuth, Lee Spector, and James Matheson. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation*, 19(5):630–643, 2014.

Samuel Holt, Zhaozhi Qian, and Mihaela van der Schaar. Deep generative symbolic regression. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=o7koEEMA1bR`.

Joey Hong, David Dohan, Rishabh Singh, Charles Sutton, and Manzil Zaheer. Latent programmer: Discrete latent codes for program synthesis. In *International Conference on Machine Learning*, pp. 4308–4318. PMLR, 2021.

Nan Jiang and Yexiang Xue. Racing control variable genetic programming for symbolic regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12901–12909, 2024.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35:10269–10281, 2022.

Pierre-Alexandre Kamienny, Guillaume Lample, Sylvain Lamprier, and Marco Virgolin. Deep generative symbolic regression with monte-carlo-tree-search. In *International Conference on Machine Learning*, pp. 15655–15668. PMLR, 2023.

William La Cava, Thomas Helmuth, Lee Spector, and Jason H Moore. A probabilistic and multi-objective analysis of lexicase selection and ε-lexicase selection. *Evolutionary Computation*, 27 (3):377–402, 2019.

William La Cava, Bogdan Burlacu, Marco Virgolin, Michael Kommenda, Patryk Orzechowski, Fabrício Olivetti de França, Ying Jin, and Jason H Moore. Contemporary symbolic regression methods and their relative performance. *Advances in neural information processing systems*, 2021(DB1):1, 2021.

Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*, pp. 5979–5989. PMLR, 2021.

Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K Petersen. A unified framework for deep symbolic regression. *Advances in Neural Information Processing Systems*, 35:33985–33998, 2022.

Wenqiang Li, Weijun Li, Linjun Sun, Min Wu, Lina Yu, Jingyi Liu, Yanjie Li, and Songsong Tian. Transformer-based model for symbolic regression via joint supervised learning. In *The Eleventh International Conference on Learning Representations*, 2022.

Wenqiang Li, Weijun Li, Lina Yu, Min Wu, Linjun Sun, Jingyi Liu, Yanjie Li, Shu Wei, Deng Yusong, and Meilan Hao. A neural-guided dynamic symbolic network for exploring mathematical expressions from data. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=IejxxE9DO2`.

Jiacheng Liu and Siqi Guo. Symbolic regressions in non-physical systems, 2023. URL `https://openreview.net/forum?id=RuCQRXk7a7G`.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Skq89Scxx`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. Rethinking symbolic regression datasets and benchmarks for scientific discovery. *Journal of Data-centric Machine Learning Research*, 2022.

Kazem Meidani, Parshin Shojaee, Chandan K. Reddy, and Amir Barati Farimani. SNIP: Bridging mathematical symbolic and numeric realms with unified pre-training. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=KZSEgJGPxu`.

Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD explorations newsletter*, 3(1):27–32, 2001.

Alberto Moraglio, Krzysztof Krawiec, and Colin G Johnson. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature-PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I 12*, pp. 21–31. Springer, 2012.

Terrell Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P Santiago, Brenden K Petersen, et al. Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. *Advances in Neural Information Processing Systems*, 34:24912–24923, 2021.

Ji Ni, Russ H Drieberg, and Peter I Rockett. The use of an analytic quotient operator in genetic programming. *IEEE Transactions on Evolutionary Computation*, 17(1):146–152, 2012.

Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10:1–13, 2017.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Md Rizwan Parvez, Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. Retrieval augmented code generation and summarization. *arXiv preprint arXiv:2108.11601*, 2021.

Tomasz P Pawlak, Bartosz Wieloch, and Krzysztof Krawiec. Semantic backpropagation for designing search operators in genetic programming. *IEEE Transactions on Evolutionary Computation*, 19(3):326–340, 2014.

Christian Raymond, Qi Chen, Bing Xue, and Mengjie Zhang. Adaptive weighted splines: A new representation to genetic programming for symbolic regression. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 1003–1011, 2020.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pp. 4442–4450. Pmlr, 2018.

Parshin Shojaee, Kazem Meidani, Amir Barati Farimani, and Chandan Reddy. Transformer-based planning for symbolic regression. *Advances in Neural Information Processing Systems*, 36, 2024a.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. Llm-sr: Scientific equation discovery via programming with large language models. *arXiv preprint arXiv:2404.18400*, 2024b.

Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic physics learner: Discovering governing equations via monte carlo tree search. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=ZTK3SefE8_Z`.

YuHong Sun, Zhangyue Yin, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Hui Zhao. Benchmarking hallucination in large language models based on unanswerable math word problem. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 2178–2188, Torino, Italia, May 2024. ELRA and ICCL. URL `https://aclanthology.org/2024.lrec-main.196`.

Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *Advances in Neural Information Processing Systems*, 33:4860–4871, 2020.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=1PL1NIMMrw`.

Zora Zhiruo Wang, Akari Asai, Xinyan Velocity Yu, Frank F Xu, Yiqing Xie, Graham Neubig, and Daniel Fried. Coderag-bench: Can retrieval augment code generation? *arXiv preprint arXiv:2406.14497*, 2024.

Yilong Xu, Yang Liu, and Hao Sun. Reinforcement symbolic regression machine. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=PJVUWpPnZC`.

Hengzhe Zhang, Aimin Zhou, Hong Qian, and Hu Zhang. PS-Tree: A piecewise symbolic regression tree. *Swarm and Evolutionary Computation*, 71:101061, 2022.

Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, and Mengjie Zhang. Modular multi-tree genetic programming for evolutionary feature construction for regression. *IEEE Transactions on Evolutionary Computation*, 2023a.

Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, and Mengjie Zhang. A semantic-based hoist mutation operator for evolutionary feature construction in regression. *IEEE Transactions on Evolutionary Computation*, 2023b.

Michael Zhang, Samuel Kim, Peter Y Lu, and Marin Soljačić. Deep learning and symbolic regression for discovering parametric equations. *IEEE Transactions on Neural Networks and Learning Systems*, 2023c.